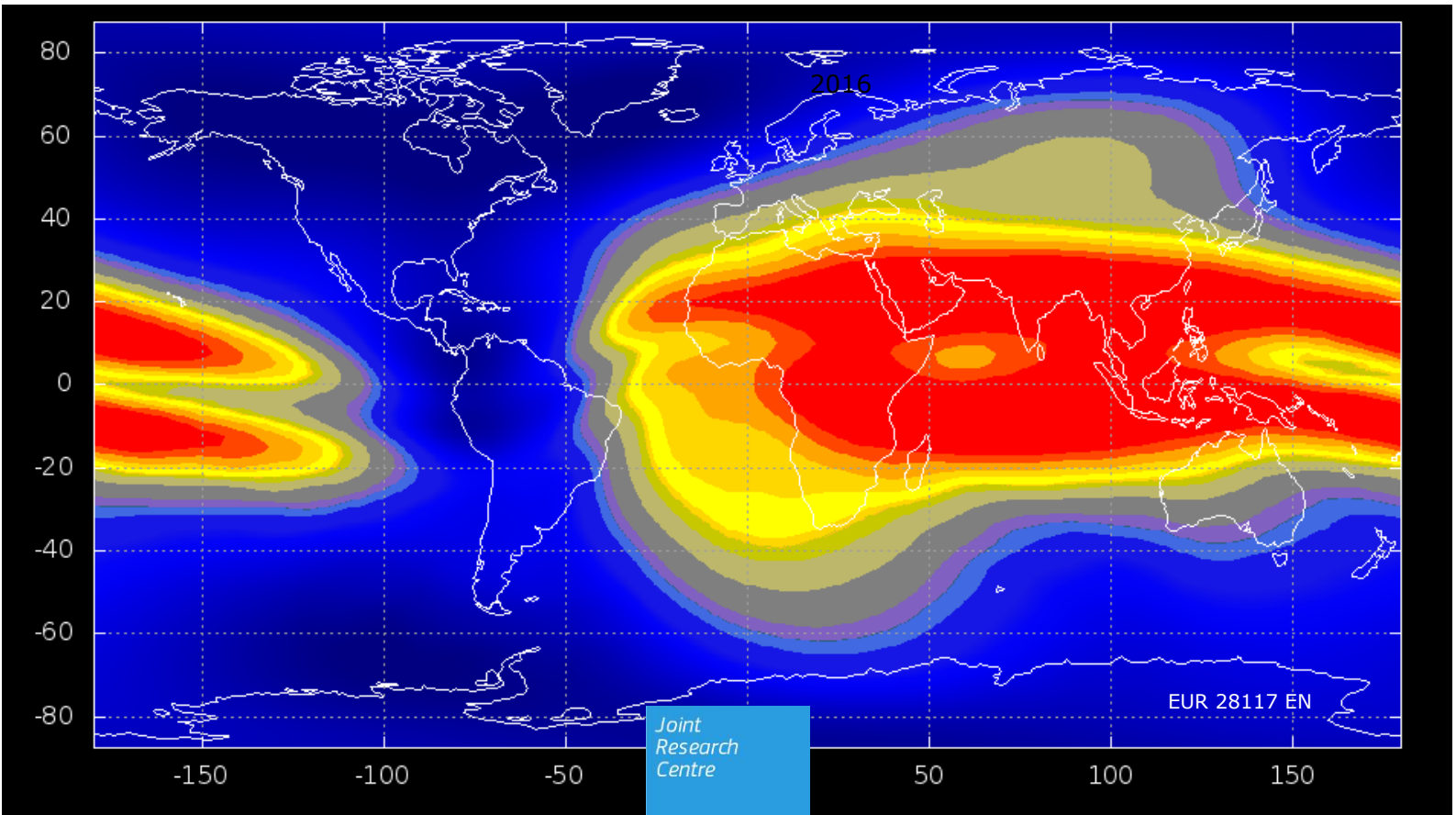
European Commission

JRC TECHNICAL REPORTS

Software prototype for the Galileo Ionospheric Correction Model Progress Report

Aragon Angel M.A.

Zürn M.



JRC Science Hub
https://ec.europa.eu/jrc

JRC103141

EUR 28117 EN

Print	ISBN 978-92-79-62024-9	ISSN 1018-5593	doi:10.2788/835895
PDF	ISBN 978-92-79-62023-2	ISSN 1831-9424	doi:10.2788/63249

Luxembourg: Publications Office of the European Union, 2016

How to cite: Aragon Angel M.A., Zürn M; Software prototype for the Galileo Ionospheric Correction Model: Progress Report; EUR 28117 EN; doi:10.2788/835895

Table of contents

Abstract

The document describing the particular ionospheric model developed for the Galileo satellite navigation system has been officially released [1]. Its publication allows GNSS receiver manufacturers to start the implementation of the specific algorithm targeted for their Galileo related products in order to be compliant with the Galileo system. Some of such implementations have already been supported by Unit G05 (now E.2) in the framework of an on-going support activity under the EU GNSS Programmes with DG GROW and GSA, responding to inquiries from the receiver manufacturers Broadcom and Javad. The only way to support the development of the new Galileo ionospheric correction in an efficient and timely manner is to have our own developed model as an official Galileo Programme reference (i.e. the European Commission) product to assess and help correcting external implementations, which will definitely have a direct impact on industry and, in return, society. Once the JRC NeQuickG product is available, further developments have been already foreseen: optimize the algorithm and tailor it to the needs of the various users, notably in terms of accuracy.

This particular report includes a description of the JRC NeQuick source code (JRC NeQuickG) and comments on the challenges faced to implement it. Emphasis on the System Architecture is put in order to help any potential follow up of this project in the near future.

1 Introduction

With the launch programme of Galileo, the European global navigation satellite system (GNSS) is in full swing, the policy makers as well as industry have turned their attention to ensure market readiness with SATNAV receivers able to use Galileo signals. This requires providing necessary standardised tools for receiver manufacturers to produce Galileo-compatible satnav receivers.

The objective of this project is to help the industry in manufacturing Galileo-compatible receivers. Specifically, the results of this project will help receiver manufacturers in ensuring that their receiver designs are based on a correct implementation of Galileo's standard model for ionospheric error correction. Ionospheric error correction allows a receiver to deal with the effect of unpredictable disturbances in the satellite navigation signals so as to achieve desired accuracy in calculating its position coordinates. Galileo's standard model for ionospheric error correction is called NeQuickG which is mathematically very complex. Its correct implementation is very difficult and results need to be validated against a reference implementation to assure compliance with system specifications. This is a fundamental technical requirement for the industry to be able to deliver high-precision positioning capability expected from Galileo. A Commission document with the detailed specifications of NequickG has been published recently. Every single Galileo receiver must implement this model to achieve the nominal positioning accuracy for Galileo.

Our experience with leading players in the GNSS industry (e.g. Broadcom and Javad) in achieving NeQuickG compliance of their Galileo receivers has shown that even large companies with huge in-house R&D capacity need expert help in this area; for SMEs the NeQuickG conformance is nearly impossible on their own. The JRC proposes to develop a reference implementation of NeQuickG model to be made available on a portal to validate the results from third-party tools. In this way, instead of impeding growth of market for 3rd party software, JRC's reference implementation would act as a validation support tool. Alternatively, the JRC software could be licensed to software vendors to develop their own market by lowering barrier to entry.

2 Overview description of the reference documentation

The official document "Ionospheric Correction Algorithm for Galileo Single Frequency Users" [1], available at www.gsc-europa.eu/electronic-library/programme-references, describes the particular ionospheric model developed for the Galileo satellite navigation system. Note than more than 200 mathematical formulas are used in this description. Its publication has allowed GNSS receiver manufacturers to start the implementation of the specific algorithm targeted for their Galileo related products in order to be compliant with the Galileo system.

The aforementioned document contains the description of the specific ionospheric model for Galileo (from here onwards referred to as NeQuickG) along with the complementary files required to run it:

- Twelve CCIR files needed for the calculation of foF2 and M(3000)F2, the critical frequency at the F2 layer and the propagation factor, respectively.
- MODIP file needed to calculate the Modified Dip Latitude (MODIP) at the receiver's location.

A qualitative description of the model is provided in Annex E. Moreover, Input/Output verification data are provided for different levels of solar activity (low, medium and high): 108 examples are given in total. They are grouped into 3 sections. These sections refer to high, medium, and low solar activity. An appendix (Annex F in [1]) with a pseudo-code implementation is also included to guide potential implementations.

The NeQuickG model is based on NeQuick [3], which is a 3D climatological model, as opposed to single-layer ionospheric models such as the GPS model, Klobuchar [4], or the ones used in Satellite Based Augmentation Systems (SBAS). The main driver of the NeQuickG implementation is the effective ionization level, Az, as opposed to the monthly-mean sunspot number, R12, or the 10.7 cm solar radio flux, F10.7, that trigger the original NeQuick. The effective ionization level Az is calculated at the user's location following:

$$Az = a_{i0} + a_{i1}\mu + a_{i2}\mu^2$$

where the three coefficients (a_{i0}, a_{i1}, a_{i2}) are transmitted by each Galileo satellite in the F=NAV (E5a) and I=NAV (E1) navigation messages (see), combined with the MODIP latitude at the receiver's location, μ:

$$\tan(\mu) = \frac{I}{\sqrt{\cos(\varphi)}}$$

being I the true magnetic inclination and φ the geographic latitude of the receiver [5].

Parameter	Definition	Bits	Scale factor	Unit
a_{i0}	Effective Ionisation Level 1st order parameter	11	2^{-2}	sfu**
a_{i1}	Effective Ionisation Level 2nd order parameter	11*	2^{-8}	sfu**/degree
a_{i2}	Effective Ionisation Level 3rd order parameter	14*	2^{-15}	sfu**/degree^2
SF_1	Ionospheric Disturbance Flag for region 1	1	N/A	dimensionless
SF_2	Ionospheric Disturbance Flag for region 2	1	N/A	dimensionless
SF_3	Ionospheric Disturbance Flag for region 3	1	N/A	dimensionless
SF_4	Ionospheric Disturbance Flag for region 4	1	N/A	dimensionless
SF_5	Ionospheric Disturbance Flag for region 5	1	N/A	dimensionless
Total Ionosphere Correction Size		**41**		

Figure 1: Ionospheric model parameters, corresponding to Table 68 in [2]. Courtesy of the European Commission.

The rest of the inputs to run both NeQuick and NeQuickG are: month of the year, Universal Time (in hours), the positions of the navigation receiver and the transmitting satellite expressed in longitude and latitude in decimal degrees as well as their corresponding heights in meters.

The one and only output parameter of NeQuickG is the Total Electron Content i.e. the electron density along a cross section of 1 square meter. TEC is expressed in units of $10^{16}m^{-2}$ also known as TECU.

2.1 Chapter 2

In Subsection 2.5 of Chapter 2 from [1], one can find the detailed description from a theoretical point of view of the NeQuickG model. The model is presented by introducing the physical parameters that are required to build an ionospheric electron density profile, in which the model is actually anchored. The 202 equations that constitute the core of the NeQuickG model are introduced in Subsection 2.5.

2.2 Chapter 9: Annex F

Annex F (which corresponds to Chapter 9 of [1]) provides the NeQuickG Detailed Processing Model most certainly aiming at readers who intend to implement the model. The pseudo code provided in Annex F can be followed to actually implement a "homemade" version of NeQuick G.

2.3 Chapter 2 versus Chapter 9

Let us note that Annex F can be followed to implement NeQuickG in a base 0 model. Base 0 refers to the concept that the first element of a vector has the index number zero. Programming languages such as C, or Python are indeed base zero languages while Fortran is a base 1 language. Unlike annex F, the main document uses index base 1.

Moreover, variable names in Annex F have no obvious correspondence to the names used in the main part of the document i.e. from Chapters 1 through Chapter 3. This infers particular difficulties for a programmer, especially owing to the fact that it is by no means trivial to transpose equations from base 0 to 1 or vice-versa. This is a major source for programming errors and misconceptions.

3 Programming phase

3.1 Roles and concepts

It was internally agreed upon the following distribution of roles:

- ✓ Dr. M. Angeles Aragon Angel has taken the role of referencing and verifying the preliminary pieces of code using the official up-to-date version on the C-code for reference purposes.
- ✓ Dr. Martin Zürn has performed the programming of new code from scratch. This approach is pragmatic because of the lack of additional reference data in [1]. Moreover, it protects the programmer from taking over pieces of code, possibly without understanding all of its implications.

3.2 Initial coding in Python

Additionally it was decided to program the first versions of new code pieces in Python as further protection against the temptation to re-use old code pieces. Furthermore, the code was based on the concepts in Chapter 2 of [1], i.e. the theoretical part of the reference document. This includes that the variables are named in a way that maintains the link to this main part of the reference document. The conversion from base 1 to base 0 code was done during the development of the Python code.

The newly developed Python modules were tested by the programmer with values made available by the auditor using the original software in an interactive process involving two persons.

3.3 Python versus C

Though Python is a modern programming language with plenty of attractive features and concepts it is obviously not a good candidate for a full implementation of NequickG. It is simply too slow for this. The structure of NequickG requires most of the functional units to be called repeatedly, most of its functions are in the inner loop while the outer loop - the integration along the ray path between navigator and satellite - is a simple task which would not justify using Python as "glue" language for different sub parts. Using Python as a glue language is very common in, for instance, Software defined radio practice.

Nevertheless, it is a smart approach to start with alpha modules in Python. The advantages are as follows:

- The actors in the project are not C programmers but scientists programming on a non-routine basis and using a wide set of different programming languages without being specialists in C. The most expensive resource is the time of the programmers. The time saved thanks to a faster progress with an easy and less bug prone language is not eaten up by a final effort to translate an already correct and fully understood module from Python to C.
- Programming development in Python is simply faster and less painful. Python follows the paradigm of dynamic typing (type declaration), this means there is no need to declare variables, this means the keyboard typing is faster and there are no troubles with forgotten declarations.
- Python does not use an overwhelming number of superfluous delimiters but the loop structure must be realized with proper indenting of the source code. This means structured programming is not a matter of virtue or discipline but it is imposed by the language.
- Modularity is extremely easy to achieve, any new module can be seen as and treated like a new library, there is no trouble with header files and function prototypes like in C.
- Python allows functions with multiple output values without the use of pointer that may often create confusion and handling errors.
- The Python motto "Batteries included" means that a myriad of functions are already readily available, including lots of vector and matrix operations.

3.4 File structure of the Python software suite

3.4.1 Input files

Table 1: Input files in Python implementation

Name of file	Content
AzCoeff.txt	File containing (ai0, ai1, ai2), which represent the solar activity
AzCo_hi.txt	File as above with a parameter example of high solar activity such as the one used in Chapter 8.1 of [1]
AzCo_mi.txt	File as above with a parameter example of medium solar activity such as the one used in Chapter 8.2 of [1]
AzCo_lo.txt	File as above with a parameter example of low solar activity such as the one used in Chapter 8.3 of [1]
modipNeQG_wrapped.txt	File containing MODIP Grid points
CCIRnn.TXT	Monthly CCIR files necessary to compute the critical frequency of the F2 Layer and the he ratio of the maximum usable frequency at a distance of 3000 km to the F2 layer critical frequency. The symbol *nn* stands for number of the month plus ten. CCIR11.TXT is for January and CCIR22.TXT is for December

These data are supplied by the original creators of the Nequick model [2], the values are not representing the ionosphere and magnetic model of today but they are kept like this for reference and test purposes. Once the model is ready for practical applications these values must be updated. Additional input such as the values from Chapter 8 are using the file suffix *.in* or no suffix while being strict ASCII files, preferably without using tabulators.

3.4.2 Python files

Table 2: Python files in Python implementation

Name of file	Content
neqlib.py	This is the name of the newly written Python library. It contains all functions and subroutines needed in any of the main programs below. The length of the file does not take away its strict modularity. Most of the function definitions could also be used as stand-alone libraries. However, some functions are only used inside other functions. The file neqlib.pyc is created by Python itself.

main.py	Python files are directly executable since Python is not a compiled language. The extension *.py* is normally used but this is just a convention without any technical meaning. There are several \main" programs for different test purposes such as Stest.py for the slant ray tests from Chapter 8 of [1]. To use any of the library functions it should be prefixed by the name of the library as usual in Python. Of course, any library name can be mapped to a shorter name. The convention is to load the library with the Python command: *import neqlib as n*

3.4.3 Output files

Some of the main files are producing screen output, others are producing ASCII files using the extension *.out*.

3.5 File structure of the C software suite

3.5.1 Input files

The input files are identical to the ones from the Python suite.

3.5.2 C specific files

Table 3: Specific C files in C implementation

Name of file	Content
neqlib.c	This le is several hundred lines long and it contains all NequickG functions
neqlib.h	C Header le which contains all constants and all function prototype used in the library code and/or any main program. This header le must be referred to in any C program using the library
main.c	Any main source code using the functions of the library. main is not really the name of a particular program but it stands for any program using NequickG functionality
main.cx	Executable version of the above
main	Executable shell script to make, compile, link, and execute the code after changes to the code. The main.cx executable should be used only if a time critical multiple

	execution is required

3.5.3 Output files

The output files follow the same principles of the Python output. C files are intended to be drop-in replacements for Python files.

3.6 Translation of Python code to C code

The translation phase is a pretty straightforward step. Whenever possible all variable names are kept exactly like the variable names in Python. Likewise the algorithms are the same. Additional changes were rare and limited to cases where Python offered particular features not present in C. Namely this happened only once when re-formatting matrices read from the above CCIR files.

4 Error reporting to reference document

The implementation process of the JRC NeQuickG is leading to the discovery of some inconsistencies that we have been come across while performing our own assessment of the theoretical and practical part of the official reference. In such document, there is Annex F devoted to the presentation of the pseudo code associated to the theoretical part, which is presented in the main part of the document. The pseudo code explanation should comply with the theoretical part in order not to lead to any misunderstanding.

At this point it should be stressed that we have no intention to interfere with ESA's work and, therefore, results presented in Annex D or Annex E have not been questioned at all.

4.1 Initial error reporting

In the following table, the compilation of errors that have been found while reviewing the document from an implementation point of view can be found. Errors, classified as minor or major according to the impact they may have, are listed. For each item, not only the error is highlighted but a way to overcome it or an alternative reference is also provided. This list pushed for the official publication of an ERRATA SHEET at the Galileo Service Centre web portal.

Table 4: List of initial errors

Source	Comment	
Section 2.5.3.1.	In the last sentences of this section it is said: "also there is an extra first and last rows phased 180 degrees in ***longitude*** to wrap the poles around." It should be: "also there is an extra first and last rows phased 180 degrees in ***latitude*** to wrap the poles around."	Minor
Section 2.5.4.3.	The extreme cases where the latitude is 90 degrees or -90 degrees have to be processed	Minor

	apart from the general case.	
Eq.5	In order to be consistent with the formulas appearing in Section 2.5.4.3. the **i index** in $stModip_{i,j}$ has to take the values **from 0 to 38 instead of 1 to 39** as stated now.	Minor
Section 2.5.5.2.	Regarding the Inputs, Solar zenith angle and Solar zenith angle at day night transition are not required. The F2 critical frequency foF2 missing. Concerning the Outputs, NmF1 is missing.	Minor
Eq. 69	**M(3000)F2_0** should be renamed to **M(3000)F2_1** in order to be accounted for in Eq. 76. Otherwise, it becomes a term that does not contribute to Eq. 76 hence leading to wrong results.	Major
Eq. 37	It does not actually reflect how foF1 calculation should be implemented. If followed as such, it leads to errors. There are more details given about how to implement it in Annex F under the subsection "Estimate foE and f0F1". The steps in Section 9.2.9.1. are the ones to be followed.	Major
Eq. 38 and Eq. 39	The condition for Eq. 38 and 39 will be updated to: If foF1 is too close to foF2, then: foF1 = 0.85*1.4*foE, if 1.4*foE>0.85*foF2 Eq. 38 The F layer maximum density NmF1 [1011 m-3] as a function of foF1 [MHz] is computed as: NmF1 = 0.124*foF12 Eq. 39	Minor
Eq. 42	It should be $fm\mathbf{3}_{i,j,k}$ instead of $fm^3{}_{i,j,k}$	Minor
Eq. 79	The current inputs are wrong. They have to be replaced by: hmF2 [km], hmE [km]	Minor
Section 2.5.5.1	**NmE [10^{11} m^{-3}]** has to be added in the outputs	Minor
Section 2.5.5.2	**khi** and **khi0** need to be removed from the inputs and add **foF2.** In the outputs, **NmF1 [10^{11} m^{-3}]** needs to be included.	Minor

Section 2.5.5.9.	**foF1 [MHz]** is missing in the list of inputs.	Minor
Section 2.5.5.9.	Before Eq. 93 there is a missing equation. It is required to include **A2=4.0*NmF1**.	Major
Section 2.5.5.9.	Eq. 94, Eq. 95 and Eq. 96 should be indented in order to emphasize the fact that they belong to an iteration that has to be performed when the if condition is followed (if foF1 >= 0.5). The sentence starting with "Then compute" should be out of the indention and at the same level as the if-conditions. See next item.	Minor
Section 2.5.5.9.	According to the text, there should be a recursive process to be repeated 5 times. The way equations are displayed there is no recursion involved. Please look at Section 9.2.9.1. for clear instructions.	Major
Section 2.5.5.10.	Missing input **Azr**.	Minor
Eq. 110	BF1top and BF1bot should be **B1top** and **B1bot**.	Minor
Section 2.5.6.1.	Eq. 111, Eq. 112 and Eq. 113 do not actually reflect how the calculation is done according to Annex F, where a condition regarding the height appears. Please follow directions given in Section 9.2.11.1. that present the correct implementation.	Major
Eq. 128	$\|x^2\|<10^{10}$ has to be replaced by $\|2*x\|<10^{-10}$ or $\|x\|<5*10^{-11}$.	Minor
Eq. 174	The **negative sign** in front of the formula has to be **removed**.	Major
Eq. 175	The **negative sign** in front of the formula has to be **removed**.	Major
Section 2.5.8.2.7.	At the end of the section, when indicating the integration accuracy, for the first case:	Minor

	ε=0.001 for the integration between $\mathbf{s_0}$ and s_a it should say between $\mathbf{s_1}$ and s_a	
Eq. 165	Replace $\hat{\lambda}_1$-$\hat{\lambda}_p$ by $\hat{\lambda}_p$-$\hat{\lambda}_1$	Minor
Eq. 166	Replace $\hat{\lambda}_1$-$\hat{\lambda}_p$ by $\hat{\lambda}_p$-$\hat{\lambda}_1$	Minor
Eq. 167	Replace $\hat{\lambda}_1$-$\hat{\lambda}_p$ by $\hat{\lambda}_p$-$\hat{\lambda}_1$	Minor
Section 9.2.4.3.	In the Internal Processing return, in the description of Az, the subscript and superscript should be **i** and not **i-1**.	Minor
Section 9.2.9.1.	There is one NeqGetF2FreqFromCCIR reference misspelled as NeqGetF2Freq**s**FromCCIR.	Minor
Section 9.2.9.3.	The following elements are not part of the described implementation: harm = number of harmonics in expansion, m = rows in CCIR[], mm = cols in CCIR[], m3 = total elements in CCIR[]	Minor
Table 38	Input parameters described there do not correspond to function NeqCalcTopsideNe. They should be: dH: height of the point ptLayers: input data structure containing the properties for all the Epstein layers (pdAmp, pdPeakHeight, pdBotThick, pdTopThick)	Minor
Table 39	Input/Output parameter described there does not correspond to function NeqCalcTopsideNe. It should be: pdNmax: Maximum Ne (F2 peak)	Minor

4.2 Latest error reporting

A second batch of errors has been provided to DG for Internal Market, Industry, Entrepreneurship and SMEs, Unit J1 – EU Satellite Navigation Programmes Management during the end of April 2014. After several interchanges of information with Brussels and ESA, this has led to the upgrade of document [1], which is now Issue 1.2.

Table 5: Latest list of reported errors

Source	Comment	
1.1. Document Scope	The term "Galileo" is used to refer to system established under the European GNSS (Galileo) programme. Suggestion: The term "Galileo" is used to refer to **the** system established under the European GNSS (Galileo) programme.	Minor
Section 2.5.2.	In Table 2, Zenith angle at night-day transition should be set to: 86.23292796211615 as in the original NeQuick formulation. Otherwise the current value that appears in the documents leads to lack of accuracy in cases of high solar activity.	Major
Section 2.5.4.8.	In Eq. 28, Zenith angle at night-day transition should be set to: 86.23292796211615 as in the original NeQuick formulation. Otherwise the current value that appears in the documents leads to lack of accuracy in cases of high solar activity.	Major
Section 2.5.5.2.	After Eq. 37, one finds the following line starting with: "The F layer maximum density NmF1 …". It should say: "The F**1** layer maximum density NmF1 …"	Minor
Section 9.2.3.1.	In the expression for the calculation of sdelta, it appears 282634 but it should be 282**.**634 as indicated in Eq. 23	Major
Section 9.2.3.1.	In if2 when calculating stP2.dS = sqrt(stP2dR^2-stRay.dR^2) a dot is missing: stP2.dS = sqrt(stP2**.**dR^2-stRay.dR^2)	Minor
Section 9.2.6.1.	In the internal processing, when arriving at the first loop, it says: Loop through the G15 and K7 integration points. It should be: Loop through the **G7** and **K15** integration points.	Minor

Section 9.2.8.1.	In the internal processing, in the second line, one reads "all function NeqCalcEpstParams". It should be: "**C**all function NeqCalcEpstParams"	Minor
Section 9.2.9.1.	"Blend high and low activity cases in **ration** RR2:RR1" should be "Blend high and low activity cases in **ratio** RR2:RR1"	Minor
Section 9.2.9.1.	Calculate peak electron densities for the layers: Reference to peakh() is wrong. It should be NeqCalcF2PeakHeight	
Section 9.2.9.1.	In the alternative to the if1, else1 October to May should be else1 October to **March**	
Section 9.2.9.3.	In Note1 it says that those terms are to be set to zero if found to be $\leq 10^{-30}$. It should say if their **absolute value** is found to be $\leq 10^{-30}$ and it applies to the first sum in the foF2 or M(3000)F2 expressions.	Minor
Section 9.2.12.2.	The argument in the exponential function should be dPower instead of Power in the second case of the Internal Processing	Minor
Section 9.2.9.1.	According to the document, Chi0=86.23. Nevertheless, that leads to lack of accuracy in cases of high solar activity. Therefore, Chi0 should be set to: Chi0=86.23292796211615 as in the original NeQuick formulation.	MAJOR
Section 9.2.11.1.	It is mandatory to put the absolute value in the expressions of arg appearing in this section. Concretely, in the two denominators of the arg expression: 1+f2*fabs(h_0-PeakHeight[F2]) 1+f2*fabs(dHH-PeakHeight[F2])	MAJOR

5 Conclusion

In the realization of the Proof of Concept "Software Prototype for the Galileo Ionospheric Correction Model", which aims at the implementation of a JRC (in-house) Galileo Ionospheric corrections software, errors in the official documentation to be used as benchmark have been found and corrected after having timely reported them to the EU Satellite Navigation Programme Directorate in DG GROW. This is a critical issue since the target audience for such document is the receivers' manufacturer industry. Errors or misleading information in the reference documents will directly translate into losses of time and money and the lack of confidence in the institutions promoting such documents.

The internal efforts to implement our own software distribution have had already direct outcomes:

- An errata sheet, which has been produced highlighting errors found in the publicly available documentation, while proposing solutions and/or alternatives to overcome them. At this point, these solutions are temporary while waiting for the proper framework to discuss with ESA and seek common agreement for an updated revision of the official document. The errata sheet has been officially published by the European Commission at the Galileo Service Center (GSC) portal as document ERRATA SHEET #1.
- An update of document [1], Issue 1.2, officially published by the European Commission at the GSC portal. The publication of this new issue has led to the removal of Issue 1.1 and the ERRATA SHEET #1.

References

[1] N.N.,European GNSS (Galileo) Open Service - Correction algorithm for Galileo Single Frequency Users, Issue 1.1, ISBN 978-92-44700-6, doi: 10 2873/723786, June 2015.

[2] European Commission, "European GNSS (Galileo) Open Service Signal In Space Interface Control Document, Issue 1.1", September 2010.

[3] G. Di Giovanni and S. M. Radicella, "An analytical model of the electron density profile in the ionosphere," Advances in Space Research, vol. 10, pp. 27–30, 1990.

[4] J. A. Klobuchar, "Ionospheric time-delay algorithm for single-frequency gps users ionospheric time-delay algorithm for single-frequency gps users," IEEE Transactions on aerospace and electronic systems, vol. AES- 23, N. 3, pp. 325–331, 1987.

[5] Rawer, K., "Propagation of decameter waves (HF-band), in Meteorological and Astronomical Influences on Radio Wave Propagation", Landmark, Ed. B. Pergamon Press, 1963.

List of figures

List of tables

LB-NA-28117-EN-C

JRC Mission

As the science and knowledge service of the European Commission, the Joint Research Centre's mission is to support EU policies with independent evidence throughout the whole policy cycle.

EU Science Hub
ec.europa.eu/jrc

@EU_ScienceHub
EU Science Hub - Joint Research Centre
Joint Research Centre
EU Science Hub

Publications Office

doi:10.2788/835895
ISBN 978-92-79-62024-9